Package: countdown (via r-universe)

November 10, 2026

Title A Countdown Timer for HTML Presentations, Documents, and Web Apps

Version 0.6.0.9000

Description A simple countdown timer for slides and HTML documents written in 'R Markdown' or 'Quarto'. Integrates fully into 'Shiny' apps. Countdown to something amazing.

License MIT + file LICENSE

URL https://pkg.garrickadenbuie.com/countdown/, https://github.com/gadenbuie/countdown

BugReports https://github.com/gadenbuie/countdown/issues

Imports htmltools, prismatic (>= 1.1.0), utils

Suggests rmarkdown, shiny, testthat (>= 3.0.0)

Config/Needs/website callr, xaringan, xaringanthemer, xaringanExtra

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Repository https://gadenbuie.r-universe.dev

Date/Publication 2025-11-15 14:20:33 UTC

RemoteUrl https://github.com/gadenbuie/countdown

RemoteRef main

RemoteSha 7a5f2c060cc2f222791260b389495c5d3eb02dcb

RemoteSubdir r

Contents

countdown 2
countdown_action 7
countdown_app 7
countdown_shiny_example 8
countdown_update 9

Index **11**

---

countdown *Countdown Timer*

Description

Creates a countdown timer using HTML, CSS, and vanilla JavaScript, suitable for use in web-based presentations, such as those created by `xaringan::infinite_moon_reader()`.

Usage

```
countdown(
  minutes = 1L,
  seconds = 0L,
  ...,
  id = NULL,
  class = NULL,
  style = NULL,
  play_sound = FALSE,
  bottom = if (is.null(top)) "0",
  right = if (is.null(left)) "0",
  top = NULL,
  left = NULL,
  warn_when = 0L,
  update_every = 1L,
  blink_colon = update_every > 1L,
  start_immediately = FALSE,
  font_size = NULL,
  margin = NULL,
  padding = NULL,
  box_shadow = NULL,
  border_width = NULL,
  border_radius = NULL,
  line_height = NULL,
  color_border = NULL,
  color_background = NULL,
  color_text = NULL,
  color_running_background = NULL,
  color_running_border = NULL,
  color_running_text = NULL,
  color_finished_background = NULL,
  color_finished_border = NULL,
  color_finished_text = NULL,
  color_warning_background = NULL,
  color_warning_border = NULL,
  color_warning_text = NULL
)
```

```
countdown_fullscreen(
  minutes = 1,
  seconds = 0,
  ...,
  class = NULL,
  start_immediately = FALSE,
  font_size = "30vw",
  border_width = "0",
  border_radius = "0",
  margin = "0",
  padding = "0",
  top = 0,
  right = 0,
  bottom = 0,
  left = 0
)

countdown_style(
  font_size = "3rem",
  margin = "0.6em",
  padding = "10px 15px",
  box_shadow = "0px 4px 10px 0px rgba(50, 50, 50, 0.4)",
  border_width = "0.1875 rem",
  border_radius = "0.9rem",
  line_height = "1",
  color_border = "#ddd",
  color_background = "inherit",
  color_text = "inherit",
  color_running_background = "#43AC6A",
  color_running_border = prismatic::clr_darken(color_running_background, 0.1),
  color_running_text = NULL,
  color_finished_background = "#F04124",
  color_finished_border = prismatic::clr_darken(color_finished_background, 0.1),
  color_finished_text = NULL,
  color_warning_background = "#E6C229",
  color_warning_border = prismatic::clr_darken(color_warning_background, 0.1),
  color_warning_text = NULL,
  .selector = ":root"
)
```

Arguments

minutes	The number of minutes for which the timer should run. This value is added to seconds.
seconds	The number of seconds for which the timer should run. This value is added to minutes.
...	Ignored by countdown(). In countdown_fullscreen(), additional arguments

are passed on to `countdown()`.

`id` A optional unique ID for the `<div>` containing the timer. A unique ID will be created if none is specified. All of the timers in a single document need to have unique IDs to function properly. Unless you have a specific reason, it would probably be best to leave this unset.

`class` Optional additional classes to be added to the `<div>` containing the timer. The `"countdown"` class is added automatically. If you want to modify the style of the timer, you can modify the `"countdown"` class or specify additional styles here that extend the base CSS.

`countdown()` provides two built-in classes:

- Use `"inline"` to create an inline, rather than absolutely-positioned, timer. This is useful for timers in prose or documents.
- Use `"no-controls"` for a timer without the up/down controls.

`style` CSS rules to be applied inline to the timer. Use `style` to override any global CSS rules for the timer. For example, to display the timer relative to the position where it is called (rather than positioned absolutely, as in the default), set `style = "position: relative; width: min-content;"`.

`play_sound` Play a sound at the end of the timer? If `TRUE`, plays the "stage complete" sound courtesy of beepr. Alternatively, `play_sound` can be a relative or absolute URL to a sound file, such as an `mp3`, `wav`, `ogg`, or other audio file type. Custom sounds are only played when `countdown` is used on a webpage or Shiny app; however, they do not work in an interactive context (creating a `countdown` in console) due to JavaScript limitations in accessing local files.

`bottom` Position of the timer within its container. By default the timer is bottom-aligned using `bottom = "0"`. If `top` is set, `bottom` defaults to `NULL`.

`right` Position of the timer within its container. By default the timer is right-aligned using `right = "0"`. If `left` is set, `right` defaults to `NULL`.

`top` Position of the timer within its container. By default `top` is unset (`NULL`).

`left` Position of the timer within its container. By default `left` is unset (`NULL`).

`warn_when` Change the countdown to "warning" state when `warn_when` seconds remain. This is achieved by adding the `warning` class to the timer when `warn_when` seconds or less remain. Only applied when greater than `0`.

`update_every` Update interval for the timer, in seconds. When this argument is greater than `1`, the timer run but the display will only update, once every `update_every` seconds. The timer will switch to normal second-by-second updating for the last two `update_every` periods.

`blink_colon` Adds an animation to the blink the colon of the digital timer at each second. Because the blink animation is handled via CSS and not by the JavaScript process that decrements the timer, so the animation may fall out of sync with the timer. For this reason, the blink animation is only shown, by default, when `update_every` is greater than 1, i.e. when the countdown time is updated periodically rather than each second.

`start_immediately`
If `TRUE`, the countdown timer starts as soon as its created (or as soon as the slides, document or Shiny app are loaded).

`font_size` The font size of the time displayed in the timer.

`margin` The margin applied to the timer container, default is `"0.5em"`.

`padding` The padding within the timer container, default is `"10px 15px"`.

`box_shadow` Shadow specification for the timer, set to `NULL` to remove the shadow.

`border_width` Width of the timer border (all states).

`border_radius` Radius of timer border corners (all states).

`line_height` Line height of timer digits text. Use this value to nudge the timer digits up or down vertically. The best value generally depends on the fonts used in your slides or document. The default value is `1`.

`color_border` Color of the timer border when not yet activated.

`color_background` Color of the timer background when not yet activated.

`color_text` Color of the timer text when not yet activated.

`color_running_background` Color of the timer background when running. Colors are automatically chosen for the running timer border and text (`color_running_border` and `color_running_text`, respectively) from the running background color.

`color_running_border` Color of the timer border when running.

`color_running_text` Color of the timer text when running.

`color_finished_background` Color of the timer background when finished. Colors are automatically chosen for the finished timer border and text (`color_finished_border` and `color_finished_text`, respectively) from the finished background color.

`color_finished_border` Color of the timer border when finished.

`color_finished_text` Color of the timer text when finished.

`color_warning_background` Color of the timer background when the timer is below `warn_when` seconds. Colors are automatically chosen for the warning timer border and text (`color_warning_border` and `color_warning_text`, respectively) from the warning background color.

`color_warning_border` Color of the timer border when the timer is below `warn_when` seconds.

`color_warning_text` Color of the timer text when the timer is below `warn_when` seconds.

`.selector` In `countdown_style()`: the CSS selector to which the styles should be applied. The default is `:root` for global styles, but you can also provide a custom class name to create styles for a particular class.

Value

A vanilla JavaScript countdown timer as HTML, with dependencies.

Functions

- `countdown()`: Create a countdown timer for use in presentations and HTML documents.
- `countdown_fullscreen()`: A full-screen timer that takes up the entire view port and uses the largest reasonable font size.
- `countdown_style()`: Set global default countdown timer styles using CSS. Use this function to globally style all countdown timers in a document or app. Individual timers can still be customized.

See Also

`countdown_app()`

Examples

```
if (interactive()) {
  countdown(minutes = 0, seconds = 42)

  countdown(
    minutes = 1,
    seconds = 30,
    left = 0,
    right = 0,
    padding = "15px",
    margin = "5%",
    font_size = "6em"
  )

  # For a stand-alone full-screen countdown timer, use countdown_fullscreen()
  # with default parameters.
  countdown_fullscreen(1, 30)

  # For xaringan slides, use percentages for `margin` to set the distance from
  # the edge of the slide and use `font_size` to adjust the size of the digits.
  # If you need to nudge the text up or down vertically, increase or decrease
  # `line_height`.
  countdown_fullscreen(
    minutes = 0,
    seconds = 90,
    margin = "5%",
    font_size = "8em",
  )

  # To position the timer "inline" in R Markdown documents,
  # use the `style` argument on each timer:
  countdown(1, 30, style = "position: relative; width: min-content;")
}
```

countdown_action *Perform a Countdown Timer Action in a Shiny App*

Description

Performs an action in a countdown timer dynamically in a Shiny app via server logic. You can start, stop, reset, or bump time time (when the timer is running) up or down. See `countdown_shiny_example()` for an example app demonstrating the usage of `countdown_action()`.

Usage

```
countdown_action(
  id,
  action = c("start", "stop", "reset", "bumpUp", "bumpDown"),
  session = NULL
)
```

Arguments

`id`	A character vector with one or more `id` values for timers created with `countdown()` or `countdown_fullscreen()`. Be sure to set the `id` value when creating the timer.
`action`	The action to perform, one of `"start"`, `"stop"`, `"reset"`, `"bumpUp"`, or `"bumpDown"`.
`session`	The reactive `session` object for the current Shiny session. In general, only required for expert or unusual use cases.

Value

Invisibly returns the `id` of the updated countdown timer(s).

See Also

Other Shiny functions: `countdown_app()`, `countdown_shiny_example()`, `countdown_update()`

countdown_app *Launch Countdown Shiny App*

Description

Launches a full screen, interactive countdown timer as a shiny-package app.

Usage

```
countdown_app(...)
```

Arguments

...	Arguments passed on to `shiny::runApp`
	`port` The TCP port that the application should listen on. If the `port` is not specified, and the `shiny.port` option is set (with `options(shiny.port = XX)`), then that port will be used. Otherwise, use a random port between 3000:8000, excluding ports that are blocked by Google Chrome for being considered unsafe: 3659, 4045, 5060, 5061, 6000, 6566, 6665:6669 and 6697. Up to twenty random ports will be tried.
	`launch.browser` If true, the system's default web browser will be launched automatically after the app is started. Defaults to true in interactive sessions only. The value of this parameter can also be a function to call with the application's URL.
	`host` The IPv4 address that the application should listen on. Defaults to the `shiny.host` option, if set, or `"127.0.0.1"` if not. See Details.
	`workerId` Can generally be ignored. Exists to help some editions of Shiny Server Pro route requests to the correct process.
	`quiet` Should Shiny status messages be shown? Defaults to FALSE.
	`test.mode` Should the application be launched in test mode? This is only used for recording or running automated tests. Defaults to the `shiny.testmode` option, or FALSE if the option is not set.

Value

Runs the countdown timer Shiny app in the current R session.

See Also

Other Shiny functions: `countdown_action()`, `countdown_shiny_example()`, `countdown_update()`

Examples

```
if (interactive()) {
  countdown_app()
}
```

---

countdown_shiny_example

Example Countdown Shiny App

Description

An example app that demonstrates the ways that countdown timers can be integrated into Shiny apps.

Usage

```
countdown_shiny_example(display.mode = c("showcase", "normal", "auto"))
```

Arguments

display.mode	The mode in which to display the application. If set to the value "showcase", shows application code and metadata from a DESCRIPTION file in the application directory alongside the application. If set to "normal", displays the application normally. Defaults to "auto", which displays the application in the mode given in its DESCRIPTION file, if any.

Value

Runs the example Shiny app in the current R session.

See Also

Other Shiny functions: countdown_action(), countdown_app(), countdown_update()

Examples

```
if (interactive()) {
  countdown_shiny_example()
}
```

---

countdown_update *Update a Countdown Timer in a Shiny App*

Description

Updates the settings of a countdown timer dynamically in a Shiny app via server logic. See countdown_shiny_example() for an example app demonstrating the usage of countdown_update().

Usage

```
countdown_update(
  id,
  ...,
  minutes = NULL,
  seconds = NULL,
  warn_when = NULL,
  update_every = NULL,
  blink_colon = NULL,
  play_sound = NULL,
  session = NULL
)
```

Arguments

id	A character vector with one or more id values for timers created with countdown() or countdown_fullscreen(). Be sure to set the id value when creating the timer.
...	Ignored, but included for future compatibility.
minutes	The number of minutes for which the timer should run. This value is added to seconds.
seconds	The number of seconds for which the timer should run. This value is added to minutes.
warn_when	Change the countdown to "warning" state when warn_when seconds remain. This is achieved by adding the warning class to the timer when warn_when seconds or less remain. Only applied when greater than 0.
update_every	Update interval for the timer, in seconds. When this argument is greater than 1, the timer run but the display will only update, once every update_every seconds. The timer will switch to normal second-by-second updating for the last two update_every periods.
blink_colon	Adds an animation to the blink the colon of the digital timer at each second. Because the blink animation is handled via CSS and not by the JavaScript process that decrements the timer, so the animation may fall out of sync with the timer. For this reason, the blink animation is only shown, by default, when update_every is greater than 1, i.e. when the countdown time is updated periodically rather than each second.
play_sound	Play a sound at the end of the timer? If TRUE, plays the "stage complete" sound courtesy of beepr. Alternatively, play_sound can be a relative or absolute URL to a sound file, such as an mp3, wav, ogg, or other audio file type. Custom sounds are only played when countdown is used on a webpage or Shiny app; however, they do not work in an interactive context (creating a countdown in console) due to JavaScript limitations in accessing local files.
session	The reactive session object for the current Shiny session. In general, only required for expert or unusual use cases.

Value

Invisibly returns the options sent to update the countdown timer(s).

See Also

Other Shiny functions: countdown_action(), countdown_app(), countdown_shiny_example()

Index

∗ **Shiny functions**
countdown_action, 7
countdown_app, 7
countdown_shiny_example, 8
countdown_update, 9

beepr, *4*, *10*

countdown, 2
countdown(), *3, 4*, *7*, *10*
countdown_action, 7, *8–10*
countdown_app, 7, *7*, *9, 10*
countdown_app(), *6*
countdown_fullscreen *(*countdown*)*, 2
countdown_fullscreen(), *3*, *7*, *10*
countdown_shiny_example, *7*, 8, *8*, *10*
countdown_shiny_example(), *7*, *9*
countdown_style *(*countdown*)*, 2
countdown_update, *7, 8*, 9, *9*

shiny-package, *7*
shiny::runApp, *8*

xaringan::infinite_moon_reader(), *2*